Appl Intell (2012) 36:800–808
DOI 10.1007/s10489-011-0297-9

Designing heterogeneous distributed GAs by efficiently self-adapting the migration period

Carolina Salto · Enrique Alba

Published online: 18 May 2011

Abstract This paper investigates a new heterogeneous method that dynamically sets the migration period of a distributed Genetic Algorithm (dGA). Each island GA of this multipopulation technique self-adapts the period for exchanging information with the other islands regarding the local evolution process. Thus, the different islands can develop different migration settings behaving like a heterogeneous dGA. The proposed algorithm is tested on a large set of instances of the Max-Cut problem, and it can be easily applied to other optimization problems. The results of this heterogeneous dGA are competitive with the best existing algorithms, with the added advantage of avoiding time-consuming preliminary tests for tuning the algorithm.

Keywords Distributed GAs · Heterogeneity · Migration period

1 Introduction

Since genetic algorithms (GAs) deal with a set of tentative solutions (known as the population) it is common sense to think in splitting the population into a set of collaborating subalgorithms to gain numerical and time reductions because of this structure [1]. Actually most parallel GAs found in the literature utilize some kind of spatial disposition for the individuals and then parallelize the resulting chunks in a pool of processors. Among the most widely known types of structured GAs, distributed GAs (dGAs) [3, 34] are specially popular optimization procedures and soon became an important branch of research, since they provide a faster and more efficient way of solving known and new problems [1]. Their advantage comes from the partitioning of the population into several subpopulations, each one running a separate GA, thus allowing a wide exploration of promising regions of the problem landscape. A sparse migration of individuals produces an exchange of genetic material among the subpopulations, that is actually responsible for enhancing the exploration-exploitation balance.

There exist distributed models where multiple search threads concurrently explore the solution space. These models are called *Type 3* parallel strategies in [13, 14] and *multiple walk* approaches in [15]. If each thread uses a different search procedure (a different method or different parameter settings, for example), we obtain a *Parallel Heterogeneous Metaheuristic* (PHM). The utilization of PHMs allows for a larger diversity and deeper exploration of the search space, which could hopefully lead to more accurate solutions. A taxonomy of PHMs is introduced in [28], and a survey about the state-of-the-art on parallel heterogeneous metaheuristics is given. According to this taxonomy, a *parallel homogeneous metaheuristic* (threads with the same search procedure and parameterizations) are a subtype of PHMs. Most distributed genetic algorithms proposed in the literature are members of this category [1].

Whatever a parallel distributed GA might be, it is very common in the literature to experimentally set the migration parameters, which are responsible for the structure of the algorithm and the intercommunication schema among the subpopulations (migration policy). This usually happens in the form of an offline set of runs previous to the actual utilization and evaluation of algorithms. In fact, this phase is

C. Salto (✉)
Fac. de Ingeniería, Universidad Nacional de La Pampa, Calle 110 esq. 9, General Pico, Argentina
e-mail: saltoc@ing.unlpam.edu.ar

E. Alba
Dpto. Lenguajes y Ciencias de la Computación, Universidad de Málaga, Campus de Teatinos, 29071, Málaga, Spain

costly and often unreported in articles on this topic. In any case, this increases the complexity in using regular dGAs.

Our proposal in this article is to self-adapt the migration policy, in particular the migration periods at which the individuals are exchanged among subpopulations. The resulting algorithm is then no longer a homogeneous set of GAs, but a new heterogeneous dGA (HdGA) no longer needing the traditional ad-hoc pre-tuning of migration parameters. The algorithm is composed of several subalgorithms that utilize the same optimization technique, a traditional GA, but that are using different migration periods, whose values change dynamically regarding the feedback obtained during the search (not set a priori in an ad-hoc manner). If such heterogeneous dGA could self-adapt at a low overhead and still be competitive, a new class of techniques could be ready for researchers to solve optimization problems. Thus, our motivation in this paper is to analyze the behavior of HdGA and to show that the heterogeneity could help to design powerful and robust optimization algorithms while reducing the cost of migration tuning present in homogeneous dGAs.

In order to evaluate the performance of our proposed algorithm, we will use the Max-Cut problem. Given an undirected graph with edge weights, this problem consists in finding a partition of the nodes into two subsets, such that the sum of the weights of the edges having endpoints in different subsets is maximized. This is a well-known NP-hard problem [26] and, besides its theoretical importance, has applications in several fields (see Sect. 3.1 for more details) and has been reformulated in a multitude of ways. Its choice as the case of study will impact positively and also in developing new solution methods that can influence in the solution of other optimization problems. We will consider here also existing algorithms for this problem to prove that our proposal is not only efficient but also accurate.

The outline of the paper is as follows. Section 2 presents the background to our parallel implementation of dGAs and the characteristics of the proposed heterogeneous algorithm. Section 3 formally presents the Max-Cut problem and its application to solve real world problems, while Sect. 4 gives details of the principal components of our algorithms. Section 5 presents the parameterizations used and Sect. 6, the analysis of the results from a numerical point of view. Finally, we summarize the conclusions and discuss several lines for future research in Sect. 7.

2 Heterogeneous dGAs

In this work we focus on distributed GAs (dGAs) [34], where the population is structured into smaller subpopulations (islands) relatively isolated one from the others (see Algorithm 1 for the structure of an elementary genetic algorithm (dGA$_i$)). Copies of individuals within a particular

Algorithm 1 Elementary dGA (dGA$_i$)

```
t = 0; {current generation}
initialize(P_i(t));
evaluate(P_i(t));
while (t < max_generations) do
  P'_i(t) = evolve(P_i(t)); {recombination and mutation}
  P'_i(t) = improve(P'_i(t)); {local search}
  evaluate (P'_i(t));
  P'_i(t) = send/receive individuals from dGA_j; {neighbor dGA}
  P_i(t+1) = select new population from P_i(t) ∪ P'_i(t);
  t = t + 1;
end while
```

subpopulation P_i (where i is the identifier of an island) can occasionally migrate to another one. A migration policy defines the island topology, when migration occurs, which individuals are being exchanged, the synchronization among the subpopulations, and the kind of integration of exchanged individuals within the target subpopulations [9, 34]. Consequently, additional parameters controlling the migration policy are needed. This in general adds an extra setting time, that in fact authors rarely report in their studies, but that represents a considerable effort both for the researcher and for the computer. Although many parameters regulate the migration, most of them are standard and widely used, like using a ring topology, sending one (or few) individuals, replacing the local worst, and using asynchronous communication to get larger speedups. The main parameter left controlling the actual intensity of the interaction that attract authors is the migration period, that is the number of generations in every subpopulation between two successive exchanges of individuals with its neighbor subpopulations. Typically, migration intervals are specified as a predefined constant number of generations. The setting of an appropriate frequency period requires considering several values, from low to high periods, in a factorial-analysis-kind-of study, carrying out the corresponding previous experimentation, thus analyzing which one of those periods will be finally used in the actual solver for the problem. This task is computationally expensive and the appropriate period depends of the characteristics of the problem to be solved. In a novel approach, we propose here a different strategy: a self-adapting dGA that automatically defines this period at a low cost. Our technique can be used by any dGA to solve an arbitrary problem and then the pre-tuning time will be minimized.

The new heterogeneous proposal consists in that each population dynamically adjusts the migration period as the genetic search proceeds. A major advantage of this adaptive criteria lies in that it is not necessary to set it to any ad-hoc value. Also, we focus in reducing the overhead to a minimum, so as to finally allow savings in the numerical

Algorithm 2 Pattern for our dynamic adaptive criteria

```
if avg_g Δf̄ < (1 + ε) · avg_{g−1} Δf̄ then
    mig_period = mig_period × 2;
else if avg_g Δf̄ > (2 − ε) · avg_{g−1} Δf̄ then
    mig_period = mig_period ÷ 2;
end if
```

effort which can be used later to solve the actual optimization problem. According to the taxonomy proposed in [28], our proposal is a Parallel Heterogeneous Metaheuristic using the same search method (a GA) but with different configurations, in particular different parameter settings (migration period).

The criterion to modify the migration period (mig_period) is a function of the average fitness of the population. Following the ideas presented in [2], we define $\Delta\overline{f}_g$ as the difference between the average fitness values in generation t and $t-1$: $\Delta\overline{f}_g = \overline{f}_t - \overline{f}_{t-1}$. We calculate the average of this difference during two consecutive migration steps (g and $g-1$); the average is denoted as $\text{avg}_g\Delta\overline{f} = \sum_{t=1}^{\text{mig\_period}} \Delta\overline{f}_t/2$. The algorithm will change the period to its closer larger power of two value if the difference $\text{avg}_g\Delta\overline{f}$ between two contiguous migrations (g and $g-1$) decreases at least by a factor of ϵ : $\text{avg}_g\Delta\overline{f} - \text{avg}_{g-1}\Delta\overline{f} \leq \epsilon \cdot \text{avg}_{g-1}\Delta\overline{f}$. On the contrary, the period will change to its closer smaller power of two value if that difference increases by a factor greater than $(1-\epsilon)$: $\text{avg}_g\Delta\overline{f} - \text{avg}_{g-1}\Delta\overline{f} > (1-\epsilon) \cdot \text{avg}_{g-1}\Delta\overline{f}$. Of course, migration periods assume discrete values in the range $[\text{mig\_period}_{\min}, \text{mig\_period}_{\max}]$. Algorithm 2 describes the basic adaptive pattern. This criterion is inexpensive to measure, since it checks simple conditions based on information already available in any standard GA, like the mean fitness.

We begin the execution by setting the mig_period to a power of two value in each island in the range [1..512], this value is randomly selected. Whenever the criterion is fulfilled, the adaptive search pattern performs a change in the migration period. The bounding cases are the total communication (migration period equals to one) or almost independent execution (migration period equals to 512). When the current value is equal to one and the algorithm needs to change to the next lower one, the algorithm does not make any change in the migration period at all. The same action is triggered when the current value is 512 and the algorithm needs to increase the value to the next one.

3 A standard and difficult benchmark: the Max-Cut problem

This well-known NP-hard problem consists in partitioning the set of nodes of a weighted graph $G = (V, E)$, where $V = \{1, \ldots, n\}$ is the set of nodes and $E = \{(i, j) : i, j \in V\}$ the set of edges, into two disjoint subsets S and $\overline{S} = V - S$, such that the sum of the weights of the edges from E that have one endpoint in S and the other in $\overline{S}$, is maximized. Let $w_{i,j}$ be the weight associated with edge $(i, j) \in E$. Then, the following expression must be maximized:

$$\text{cut}(S, \overline{S}) = \sum_{i \in S, j \in \overline{S}} w_{ij} \tag{1}$$

Goemans and Williamson [21] proposed an adaptation of Semidefinite Programming (SDP) to the problem, which is useful to calculate upper bounds on some graphs instances for heuristic performance comparison. Burer et al. [11] proposed for this problem a method called CirCut, which is a rank-2 relaxation heuristic that produces better solutions than previous methods in shorter computational times. Festa et al. [20] compared the CirCut with six new proposed heuristics derived from GRASP, variable neighborhood search (VNS), and path relinking (PR). The authors reported that their VNS algorithm with PR (VNSPR) obtains high quality solutions (approaching 5% of SDP bounds) but with long computational times. Martí et al. [29] proposed an Advanced Scatter Search (SS), which produces better results both in quality and computational times when compared with all previous approaches. More recently, a significant work on Max-Cut used both Simulated Annealing (SA) and Tabu Search (TS) and was proposed by Arraiz and Olivo [7]. They created a neighborhood generation procedure that balances diversity and quality. SA was able to report unseen optimal values for some instances and to match the best results reported on the literature for the rest, requiring less computational time compared to former approaches (i.e., an actual state of the art algorithm for Max-Cut). We include CirCut, VNSPR, SS, and SA in our computational analysis (see Sect. 6) in order to make a wide comparison of our proposal.

3.1 Practical issues

Although the Max-Cut problem might appear, at first glance, to be interesting in the academic field, its domain of applicability is extraordinarily broad, including VLSI design, statistical physics, data clustering, facility layouts, etc. In following paragraphs a more detailed description of each of these topics is carried out, in order to clearly state the practical implications of the problem we have selected in this article.

The most common applications come from the field of design of very large scale integrated (VLSI) circuits and statistical physics where the ground states of spin glasses in the presence of an external magnetic field are sought [5, 8]. In VLSI circuits, given an electronic circuit specified by a graph, the maximum cut defines the largest amount of data communication that can simultaneously occur in the

circuit. The highest-speed communications channel should thus span the vertex partition defined by the maximum edge cut.

Clustering analysis, used in many disciplines and applications [10, 35], is an important tool and a descriptive task seeking to identify homogeneous groups of objects based on the values of their attributes. The problem of cluster analysis can be modelled as a maximum cut problem [18]. The problem consists in partitioning a set of data points into groups of "closely related" observations. Cluster analysis can be formulated as a maximum cut problem by creating a graph that contains a node for each data point and an edge between each pair of points. The weight of the edge is determined by the relative "closeness" of the points represented by the nodes it connects.

Regarding layout facility problems, the single-row facility layout problem (SRLFP) has several interesting connections to the Max-Cut problem [6]. The SRLFP consists in arranging a given number of rectangular facilities next to each other along a line so as to minimize the total weighted sum of the center-to-center distances between all pairs of facilities. Several practical applications of the SRFLP have been identified in the literature [23, 31, 33].

Applications on the field of image segmentation can be found in recent years [19, 30, 32]. Several methods using a tree structured framework have been applied to high dimensional data, such as hyperspectral images, using the Max-Cut problem as a tool for constructing that hierarchical tree [24, 27]. Chen et al. [12] proposed a method, called Support Vector Machines (SVM), that utilizes a tree structure framework and solves a series of Max-Cut problems to perform the unsupervised class decomposition.

Other applications can be found in the area of combinatorial optimization where many geometric reformulations of the Max-Cut problem are possible based on the incidence vector representation. Examples here include linear programming over the cut polytope and unconstrained quadratic 0–1 programming [16, 17].

As stated in previous paragraphs, the Max-Cut problem has several and interesting applications in many practical or theoretical fields, so it is of interest to develop new techniques to provide better quality solutions to this problem, as we do in this paper. Therefore, our methods can contribute to the solution on the applications of the Max-Cut problem previously detailed.

4 Heterogeneous dGA and Max-Cut

In this section we describe the principal components of our proposed HdGA to solve Max-Cut.

Representation A solution to this problem can be represented as a binary vector $x = (x_1, x_2, \ldots, x_n)$ of length n, where x_i corresponds to a node. Each vector encodes a partition of the nodes, with $x_i = 1$ meaning $x_i \in S$ and $x_i = 0$ meaning $x_i \in \overline{S}$.

Initialization The algorithm creates several initial solutions using the constructive heuristic proposed by Kahruman et al. [25], called SG3. This method is fast and generates solutions with considerable quality and diversity. SG3 was also chosen by Arraiz and Olivo [7] to be the constructive heuristic for their SA and TS and seems to be a key component in all modern well-performing algorithms for Max-Cut.

Local search heuristic After genetic operators (crossover and mutation), the newly created individuals undergo a local optimization procedure with a certain probability. When applied on a solution x, the procedure starts by creating a set T of all nodes i (for $i = 1, \ldots, n$). The iterative procedure randomly selects a node i from the set T and then changes that node from one subset to the other in x according to the following rules:

if $x_i = 0$ and $\sigma_S(i) - \sigma_{\overline{S}}(i) > 0$ **then** $x_i = 1$

if $x_i = 1$ and $\sigma_{\overline{S}}(i) - \sigma_S(i) > 0$ **then** $x_i = 0$

where, for each node $i : 1, \ldots, n, \sigma_S(i) = \sum_{j \in S} w_{ij}$ and $\sigma_{\overline{S}}(i) = \sum_{j \in \overline{S}} w_{ij}$ [20], $\sigma_S(i) - \sigma_{\overline{S}}(i)$ represents the change in the objective function associated with moving a node i from one subset of the cut to the other. All possible moves of a solution are investigated by making a single pass through all the nodes. The current solution is replaced by the improved one. The pseudo-code of the local search procedure is given in Algorithm 3. This algorithm is an efficient variation of the local search phase proposed by Festa et al. [20] used in their GRASP method ($O(n^2)$ complexity), since our local search procedure makes one single pass through all the nodes, i.e., $O(n)$.

Table 1 Max-Cut instance description

Instances	|V|	Density (%)
G1–G2–G3	800	6.12
G11–G12–G13	800	0.63
G14–G15–G16	800	1.58
G22–G23–G24	2000	1.05
G32–G33–G34	2000	0.25
G35–G36–G37	2000	0.64
G43–G44–G45	1000	2.10
G48–G49–G50	3000	0.17

5 Experimental setup

The first step in the experimentation consists of exploring the effect of some operators that effect the subpopulations of a dGA. The following paragraph explains the set of parameters used in the experimentation for the homogeneous dGA, which were selected after an empirical parameter tuning process.

The global population of all evaluated models is composed of 512 individuals and there are 16 islands with 32 individuals each (μ). The maximum number of generations is fixed to 6500. Each parent is selected by a binary tournament. In every island, the number λ of created offspring is 32 at each iteration. The uniform crossover is applied with a probability of 0.65, the bit-flip mutation is applied to all new generated individuals with an intensity of $1/|V|$ per bit, and the local search procedure is applied with a probability of 0.2. The next population is build up from the $(\mu + \lambda)$ individuals using fitness proportional selection. The distribution scheme of islands is based on a unidirectional ring topology. The communication is asynchronous for efficiency. A copy of the best individual is sent to the neighboring subpopulation, while the target island selects the worst individual to be replaced with the incoming one (only if it has better or equal fitness).

The algorithms are implemented in MALLBA [4], a C++ software library. Our computing system is a cluster of 8 machines with AMD Phenom8450 Triple-core Processor at 2 GHz with 2 GB of RAM, linked by Gigabit, under Linux with 2.6.27-4 GB kernel version. Each island is physically run on a separate processor.

For our experiments we used the set of instances generated by Helmberg and Rendl [22] (publicly available at http://www.stanford.edu/yyye/yyye/Gset/). In particular, we use a set of widely used graphs. They consist of toroidal, planar and randomly generated graphs of varying sizes and densities, with weights taking values 1, 0, or −1. The size of the graphs (|V|) varies from 800 to 3000 nodes meanwhile the density fluctuates from 0.17% to 6.12% (see Table 1 for more details). Festa et al. [20], Martí et al. [29], and Arráiz et al. [7] all used these graphs in their experiments, so it is a convenient election for comparison purposes.

Algorithm 3 Local Search

```
T ← V;
while ( T ≠ ∅) do
  i ← random vertex in T;
  if i ∈ S and σ_S(i) − σ_S̄(i) > 0 then
    S ← S \ {i};
    S̄ ← S̄ ∪ {i};
  end if
  if i ∈ S̄ and σ_S̄(i) − σ_S(i) > 0 then
    S̄ ← S̄ \ {i};
    S ← S ∪ {i};
  end if
  T ← T \ {i};
end while
```

6 Analysis of results

In this section we describe the results of the experiments we have made for testing our HdGAs on the Max-Cut problem. This section is divided into two parts. In a first place, a comparison of our proposed HdGA and dGAs using constant migration periods is carried out. After that, a comparison of HdGA with other methods reported in the literature to solve the Max-Cut problem is conducted to determine the effectiveness of our proposal.

All the results provided here are the mean of 30 independent runs. We have performed an analysis of variance of the results to obtain meaningful conclusions. When the results happen to follow a normal distribution, we use the ANOVA test to distinguish meaningful differences among the means of the results for each algorithm, with a level of significance of $\alpha = 0.05$ (to indicate a 95% confidence level). When the results do not follow a normal distribution, we use the non-parametric Kruskal-Wallis test.

6.1 Homo versus heterogeneous approaches

In this section we will compare the results produced by the proposed HdGA and a parallel homogeneous GA, so-called HomdGA (which performs the same kind of search on different sets of greedy generated individuals) in order to show the robustness of the heterogeneous dGA as an alternative search method. As we are interested in the self-adapted migration schedule, in the case of HomdGA we report a set of values for the migration period ranging from 1 (maximum coupling among islands) to 128 (fairly isolated islands), in order to characterize the effect of the migration period in the quality of the solutions obtained. Consequently, HomdGA*i* means a homogeneous approach with period *i*.

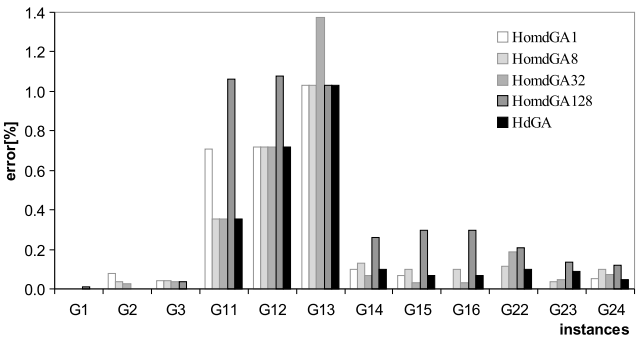


Fig. 1 Performance comparison of HdGA and HomdGA

Figure 1 shows the relative error with respect to the best-known solution in the literature for each instance. A first conclusion is that HomdGAs with low migration periods obtain better results than HomdGAs with high migration periods. The accumulated error for all instances is 2.80 for HomdGA1 (the best) and 2.57 for HdGA, that is a 8.2% lower error for HdGA. HdGA hits the best-known solution for instances G1, G2, and G3. For instances G11, G12, and G13 (representing sparse graphs with negative weights), HomdGAs and HdGA both present high errors, indicating that these instances are difficult for the algorithms. For these instances, the automatically self-adapted HdGA obtains the same error as the best Hom-dGA approach. For instances G22 and G23 the optimum values are reached by HomdGA1.

HdGA gets improvements to the best-known solution for instances G16 and G45. This is a remarkable result, since it is a long lasting very studied problem and because our true goal was to propose an algorithm that reduces pre-tuning. This result shows that it is also very accurate. In the case of instance G22, HomdGA1 obtains a fitness 13351, also better than the best solution reported in the literature (13346, until now). A similar situation is observed in instance G16, but in this case other algorithms also obtain solution values over the best-known solution 3046 (HomdGA1 3049, HomdGA8 3046, HomdGA32 3048, and HdGA 3047). Hence not only HdGA is a simpler heterogeneous version, but also a cutting edge algorithm.

Summarizing, HdGA obtains the same quality of results, or even better, than any HomdGA in 7 of the 12 instances used in this study. For the rest of the instances, the error difference is rather low and similar. These claims are corroborated by an ANOVA test: the p-values are higher than $\alpha = 0.05$ for the majority of the instances. Consequently, we have reached the proposed objective of building an alternative search method based on heterogeneous dGAs.

We want to finish this section with a short comment on the effective alleviation of the parameter tuning cost that our heterogeneous dGA provides. We must remind that no tuning of migration has been done for HdGA, while for HomdGA it was considerably hard. First of all, our heterogeneous dGA needed 26 hours of execution to obtain the results from the 30 independent executions for all the graph set addressed in this work (see Table 1). For the HomdGA, a range of values was considered for the tunable parameter, in fact we tested 4 different values for the migration period. For each experiment we have performed 30 runs, consequently we needed 120 runs to find the adequate configuration (90 runs more than HdGA). This process is repeated for every problem instance addressed in this work. In terms of time, that means approximately 104 hours of computation. Therefore, our proposed HdGA produces savings of 75% of the overall computation time compared to a

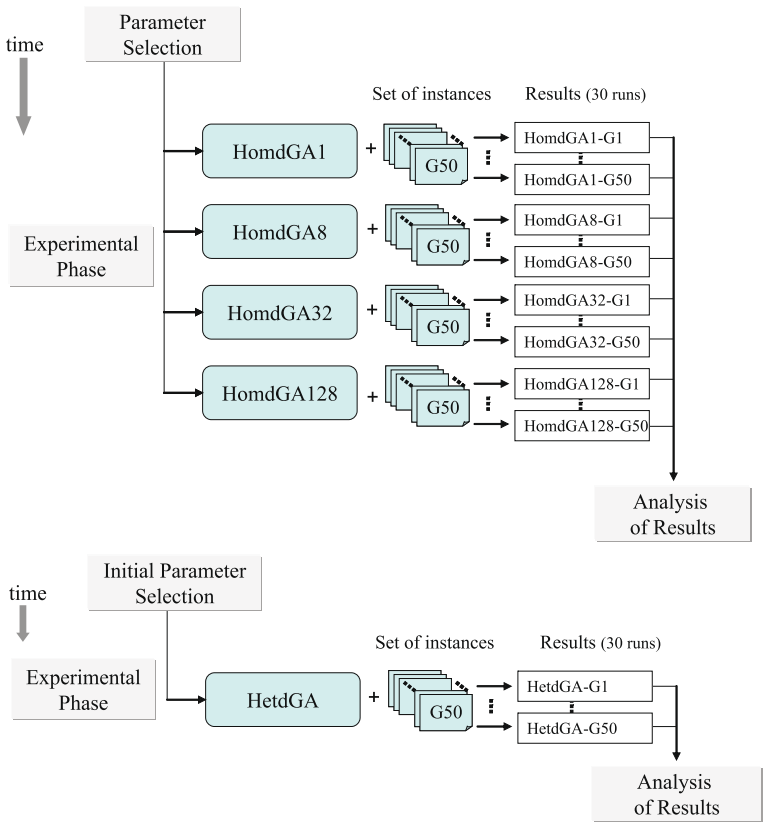


Fig. 2 Steps in the experimentation for HdGA and HomdGAs

Table 2 Performance comparison between HdGA and the best algorithms reported in literature

Inst.	HomdGA1	HdGA	CirCut	GRASP	VNSPR	SS	TS	SA	SDP
G1	**11624**	**11624**	**11624**	11540	11621	**11624**	11612	**11624**	12078
G2	11611	**11620**	11617	11567	11615	**11620**	11604	11607	12084
G3	11617	**11622**	**11622**	11551	**11622**	**11622**	11602	**11622**	12077
G11	560	562	560	552	**564**	562	562	**564**	627
G12	552	552	552	546	**556**	552	550	554	621
G13	576	576	574	572	580	578	574	**582**	645
G14	3057	3057	3058	3027	3055	**3060**	3053	**3060**	3187
G15	3047	3047	**3049**	3017	3043	**3049**	3041	3045	3169
G16	**3049***	3047	3045	3013	3043	3045	3042	3044	3172
G22	**13351***	13338	13346	13185	13295	13346	13273	13319	14123
G23	**13321**	13309	13317	13203	13290	13317	13303	**13321**	14129
G24	13307	13308	**13314**	13165	13276	13303	13285	13310	14131
G32	1394	1386	1390	1370	1396	**1398**	1382	1386	1560
G33	1350	1360	1360	1348	**1376**	1362	1354	1366	1537
G34	1356	1364	1368	1348	**1372**	1364	1364	**1372**	1541
G35	7632	7643	**7670**	7567	7635	7668	7629	7653	8000
G36	7630	7640	**7660**	7555	7632	**7660**	7627	7648	7996
G37	7637	7650	**7666**	7576	7643	7664	7642	7655	8009
G43	6651	6655	6656	6592	**6659**	6656	6658	**6659**	7027
G44	6629	6647	6643	6587	6642	6648	6641	**6650**	7022
G45	6629	**6653***	6652	6598	6646	6642	6651	6643	7020
G48	**6000**	**6000**	**6000**	**6000**	**6000**	**6000**	**6000**	**6000**	6000
G49	**6000**	**6000**	**6000**	**6000**	**6000**	**6000**	**6000**	**6000**	6000
G50	**5880**	**5880**	**5880**	5862	**5880**	**5880**	**5880**	**5880**	5988
Value sum	150460	150540	150623	149341	150441	150620	150329	150564	157743
% of SDP	4.62	4.57	4.51	5.33	4.63	4.52	4.70	4.55	

search method requiring migration period tunning. Figure 2 shows the steps in the experimentation carried out, and consequently the stated differences, in a more intuitive way.

6.2 Solution of Max-Cut

In this section, we show a comparison of the best results obtained by our heterogeneous approach and HomdGA1 with those accomplished by the state of the art methods. The results reported in Table 2 for CirCut, GRASP, VNSPR, SS, TS, and SA were taken from [7]. This table shows, for each instance and each method, the best value obtained and also the theoretical lower bound SDP. Last row gives the total sum of the best values and the mean relative percent deviation of this sum from DSP, for each method.

HdGA reports 6 of the best results known in the literature (G1, G2, G3, G48, and G49), and it also obtains unseen optimal values for G45. On the other instances, it performs relatively well, very near other methods. Moreover, an ANOVA test indicates that differences among the best values means are not significant (the p-value is higher than the significance level $\alpha = 0.05$), suggesting that the proposed algorithm presents a similar behavior to the best algorithms reported in the literature, a promising result.

Regarding wall-clock execution times, HdGA presents slightly high computation time compared to CirCut, SS, and SA, but not superior to the ones exhibited by VNSPR. The reason is that we impose a predefined effort to run our algorithms, expressed in a fixed number of generations, while the rest of the algorithms executes until a number of consecutive iterations without improvement is reached, a different stop condition. In a more in-depth analysis of the run times of HdGA, we can observed that the best values of the algorithm are obtained near the middle of the execution in the majority of the instances. This suggests that the number of generations imposed as stop condition could be decreased by half, and consequently the computational time, without affecting the quality of the final solutions.

7 Conclusions

In this paper we have presented a new heterogeneous dGA to solve the Max-Cut problem, which uses different migration periods in each subpopulation following an adaptive, automatic, and problem matching strategy. Migration periods are dynamically set based on the changes observed in the average fitness population, resulting in an inexpensive self-adaptation procedure. The proposed heterogeneous dGA are not particular to the Max-Cut problem and could be considered for future applications to other optimization problems.

We consider that the behavior of our heterogeneous dGA, as well as the homogeneous dGA, is very satisfactory since it obtains the best-known solution in most cases, or values very close to it, for all the test suite. Moreover, it has been able to improve the known best solution so far for three of the tested instances, which represents an important record in present research. Besides, as the heterogeneous algorithm dynamically determines the best period, this method helps to alleviate the setting time required, by at least 75%, to determine the best migration period in traditional dGAs, which implies a costly exploration.

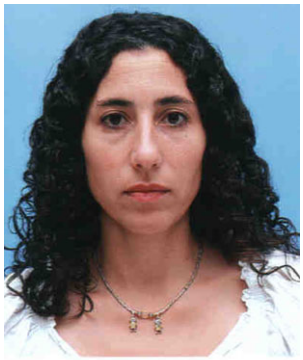
There exit however several issues that still require further analysis. The heterogeneous solver shows good results at the expense of slightly high computational time, consequently a future work will be directed to reduce the computational times required for the search. Here, specially when comparing to other distributed algorithms, it is important to quantify and compare the time needed for setting the best migration policy to be used, a step not required in the HdGA. Furthermore, we plan to extend our work to other problems and also in order to include other parameters of the migration policy to be set automatically at low cost, such as the number of individuals to be sent in each communication.

Acknowledgements This work has been partially funded by the Spanish Ministry of Science and Technology and the European FEDER under contract TIN2008-06491-C04-01 (the M* project) and project DIRICOM (P07-TIC-03044). We acknowledge the Universidad Nacional de La Pampa and the ANPCYT in Argentina from which the first author receive continuous support.

References

1. Alba E (2005) Parallel metaheuristics: a new class of algorithms. Wiley, New York
2. Alba E, Dorronsoro B (2005) The exploration/exploitation tradeoff in dynamic cellular genetic algorithms. IEEE Trans Evol Comput 9(2):126–144
3. Alba E, Troya JM (2000) Influence of the migration policy in parallel distributed GAs with structured and panmictic populations. Appl Intell 12(3):163–181
4. Alba E, Luna J, Moreno LM, Pablos C, Petit J, Rojas A, Xhafa F, Almeida F, Blesa MJ, Cabeza J, Cotta C, Díaz M, Dorta I, Gabarró J, León C (2002) MALLBA: a library of skeletons for combinatorial optimisation. LNCS, vol 2400. Springer, Berlin, pp 927–932
5. Anjos MF (2001) New convex relaxations for the maximum cut and VLSI layout problems. PhD thesis, University of Waterloo
6. Anjos M, Liers F (2010) Global approaches for facility layout and vlsi floorplanning. Handbook of semidefinite, cone and polynomial optimization: theory, algorithms, software and applications, p 32
7. Arráiz E, Olivo O (2009) Competitive simulated annealing and tabu search algorithms for the Max-Cut problem. In: GECCO '09: proceedings of the 11th annual conference on genetic and evolutionary computation. ACM, New York, pp 1797–1798
8. Barahona F, Grötschel M, Jünger M, Reinelt G (1988) An application of combinatorial optimization to statistical physics and circuit layout design. Oper Res 36(3):493–513
9. Belding TC (1995) The distributed genetic algorithms revisited. In: Proceedings of the sixth international conference on genetic algorithms, pp 114–121
10. Brdiczka O, Maisonnasse J, Reignier P, Crowley JL (2009) Detecting small group activities from multimodal observation. Appl Intell 30(1):47–57
11. Burer S, Monteiro RDC, Zhang Y (2002) Rank-two relaxation heuristics for Max-Cut and other binary quadratic programs. SIAM J Optim 12(2):503–521
12. Chen Y, Crawford M, Ghosh J (2004) Integrating support vector machines in a hierarchical output space decomposition framework. In: Proceeding of IEEE international geoscience and remote sensing symposium, vol 2, pp 949–952
13. Crainic TG, Tolouse M (2003) Parallel metaheuristics. In: Crainic TG, Laporte G (eds) Fleet management and logistics. Kluwer Academic, Dordrecht, pp 205–251
14. Crainic TG, Tolouse M (2003) Parallel strategies for metaheuristics. In: Glover FW, Kochenberger GA (eds) Handbook of metaheuristics. Kluwer Academic, Dordrecht, pp 475–514
15. Cung V, Martins SL, Ribeiro CC, Roucairol C (2003) Strategies for the parallel implementation of metaheuristics. In: Ribeiro CC, Hansen P (eds) Essays and surveys in metaheuristics. Kluwer Academic, Dordrecht, pp 263–308
16. Deza M, Laurent M (1994) Applications of cut polyhedra I. Comput Appl Math 55(2):191–216
17. Deza M, Laurent M (1994) Applications of cut polyhedra II. Comput Appl Math 55(2):217–247
18. Ding CHQ, He X, Zha H, Gu M, Simon HD (2001) A min-max cut algorithm for graph partitioning and data clustering. In: Proceedings IEEE international conference on data mining, pp 107–114
19. Du Z, Jeong Y-S, Jeong MK, Kong SG (2011) Multidimensional local spatial autocorrelation measure for integrating spatial and spectral information in hyperspectral image band selection. Appl Intell. doi:10.1007/s10489-010-0274-8
20. Festa P, Pardalos PM, Resende MGC, Ribeiro CC (2002) Randomized heuristics for the MAX-CUT problem. Optim Methods Softw 7:1033–1058
21. Goemans MX, Williamson DP (1995) Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. J ACM 42(6):1115–1145
22. Helmberg C, Rendl F (2000) A spectral bundle method for semidefinite programming. SIAM J Optim 10(3):673–696
23. Heragu SS, Kusiak A (1988) Machine layout problem in flexible manufacturing systems. Oper Res 36(2):258–268
24. Hung C-C, Kuo B-C, Chi M-H, Hsieh T-Y (2006) A comparison of hierarchical classification processes based on hyperspectral image. In: IEEE international geoscience and remote sensing symposium, pp 948–951
25. Kahruman S, Kolotoglu E, Butenko S, Hicks IV (2007) On greedy construction heuristics for the max_cut problem. Int J Comput Sci Eng 3(3):211–218
26. Karp R (1972) Reducibility among combinatorial problems. In: Miller R, Thatcher J (eds) Complexity of computer computations. Plenum, New York, pp 85–103

27. Kuo B-C, Chi M-H, Yang J-M, Yang C-W (2007) Hierarchical classification systems for hyperspectral image classification. In: IEEE international geoscience and remote sensing symposium, pp 1745–1748
28. Luna F, Alba E, Nebro AJ (2005) Parallel metaheuristics. A new class of algorithms. In: Alba E (ed) Parallel heterogeneous metaheuristics. Wiley, New York, pp 395–422
29. Martí R, Duarte A, Laguna M (2009) Advanced scatter search for the Max-Cut problem. INFORMS J Comput 21(1):26–38
30. Özyer T, Alhajj R (2009) Parallel clustering of high dimensional data by integrating multi-objective genetic algorithm with divide and conquer. Appl Intell 31(3):318–331
31. Picard J-C, Queyranne M (1981) On the one-dimensional space allocation problem. Oper Res 29(2):371–391
32. Saha S, Bandyopadhyay S (2010) Automatic MR brain image segmentation using a multiseed based multiobjective clustering approach. Appl Intell. doi:10.1007/s10489-010-0231-6
33. Suryanarayanan JK, Golden BL, Wang Q (1991) A new heuristic for the linear placement problem. Comput Oper Res 18(3):255–262
34. Tanese R (1989) Distributed genetic algorithms. In: Proceedings of the third international conference on genetic algorithms, pp 434–439
35. Verma B, Hassan SZ (2009) Hybrid ensemble approach for classification. Appl Intell 34(2):258–278



Carolina Salto received her Ph.D. degree in Computer Science from the University of San Luis, Argentina in 2009. She is currently an Assistant Professor of Computer Science at the University of La Pampa, Argentina. She has published book chapters, journal papers, and more than 40 conference papers. Her current research interests include the design and implementation of evolutionary algorithms, other metaheuristics to continuous and combinatorial problems, and parallelism applied to metaheuristics. Dr. Salto leads the 'Research Lab on Intelligent Systems'.

Enrique Alba had his degree in engineering and Ph.D. in Computer Science in 1992 and 1999, respectively, by the University of Málaga (Spain). He works as a Full Professor in this university with different teaching duties: data communications and evolutionary algorithms at graduate and master programs, respectively. Dr. Alba leads a team of 7 doctors and 8 engineers (most of them Ph.D. candidates) in the field of complex optimization. In addition to the organization of international events (IEEE IPDPS-NIDISC, IEEE MSWiM, IEEE DS-RT, ...) Dr. Alba has offered dozens doctorate courses, multiple seminars in more than 20 international institutions and has directed several research projects (5 with national funds, 5 in Europe and numerous bilateral actions). Also, Dr. Alba has directed 6 contracts for innovation and transference to the industry (OPTIMI, Tartessos, ACERINOX, ARELANCE) and at present he also works as invited professor at INRIA and the Univ. of Luxembourg. He is editor in 13 international journals and one book series of Springer-Verlag and Wiley, as well as he often reviews articles for more than 30 impact journals. He has published 38 articles in journals indexed by Thomson ISI, 17 articles in other journals, 40 papers in LNCS, and more than 100 refereed conferences. Besides that, Dr. Alba has published 11 books, 39 book chapters, and has merited 6 awards to his professional activities. Dr. Alba's H index is 22, with more than 1500 cites to his works (excluding self-cites).